Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2008 Proceedings

Americas Conference on Information Systems (AMCIS)

2008

Studying Community Dynamics with an Incremental Graph Mining Algorithm

Tanja Falkowski
University Madgeburg, tanja.falkowski@ovgu.de

Anja Barth
University Madgeburg, anja.barth@student.uni-mageburg.de

Myra Spiliopoulou
University Madgeburg, myra@iti.cs.uni-madgeburg.de

Studying Community Dynamics with an Incremental Graph Mining Algorithm

Tanja Falkowski
University Magdeburg, Germany
tanja.falkowski@ovgu.de

Anja Barth
University Magdeburg, Germany
anja.barth@student.uni-magdeburg.de

Myra Spiliopoulou
University Magdeburg, Germany
myra@iti.cs.uni-magdeburg.de

ABSTRACT

The widespread usage of the Web and later of the Web 2.0 for social interactions has stimulated scholars of different disciplines in studying electronic communities. Traditionally, communities are observed as a static phenomenon. However, they are evolving constellations, which emerge, lose members and obtain new ones and potentially, grow, coerce, split or decline. Such dynamic phenomena require the study of social networks across the time axis.

We propose the graph mining algorithm DENGRAPH for the discovery and monitoring of evolving communities. Data mining methods are successfully used for community discovery but are mostly limited to the static perspective. Taking a dynamic perspective implies the study of a stream of interactions among community members. Accordingly, our DENGRAPH is an incremental graph mining algorithm, which detects and adapts communities over time. We report on our first results in applying DENGRAPH on the social network of mail interactions of ENRON.

Keywords

Community Discovery, Community Dynamics, Graph Mining, Social Network Analysis

INTRODUCTION

In the Web, we observe a proliferation of platforms where people share information and ideas and exchange experiences. The formation of such communities and their dynamics are of interest to many disciplines for different purposes. Common to all is the need for methods that can process and effectively analyze the social structures and for models that capture the *dynamic nature* of online communities.

Communities can be studied from different perspectives. Some research groups concentrate on the spread of *influence* in social networks and have proposed methods that depict influential nodes and their neighborhoods (Domingos and Richardson, 2001; Klempe, Kleinberg and Tardos, 2003). This perspective is appropriate for studying e.g. purchase behavior in a recommendation network. Other studies concentrate rather on the *interaction* among people and are more appropriate for studying how people form groups. The community discovery method of Girvan and Newman based on the concept of *edge betweenness* in the social network is a prominent example of this perspective (Girvan and Newman 2002). In our study, we adopt this second perspective.

In comparison to the abundance of methods for community discovery in a static setting, methods for the analysis of community dynamics *for a priori unknown communities* are rather sparse, as explained in the next section. The most straightforward challenges in community dynamics are the discovery of emerging communities and their gradual adaptation as community members fluctuate. Associated with it is the weighting of old interactions in comparison to new ones. For example, if a person has been involved intensively in a community in the past, how long should her interactions be remembered, i.e. how long should she be still considered part of the community? The next challenge is the distinction between signal and noise, where signal refers to interactions that should indeed be associated with a community, while noise refers to ad-hoc activities between network members that seem but are not part of the community. Using the terminology of knowledge discovery methods, we need to discover and *incrementally adapt* communities as *clusters* over a *stream* of interactions. Since the interactions constitute a graph, we need an *incremental graph clustering algorithm* that can deal with *noisy data*.

In this study, we deal with all aforementioned challenges. We propose DENGRAPH, an incremental graph mining algorithm that discovers and adapts clusters over a network of interacting nodes. DENGRAPH is inspired by IncrementalDBSCAN (Ester et al., 1998), DENsity-based clustering algorithm that adapts clusters on batches of conventional data records. IncrementalDBSCAN has been particularly designed for noisy datasets, a desirable property for our problem at hand, but has been intended for use mainly on geographic data. A graph has very different topological properties and notion of distance than a land map, so one of the challenges resolved in DENGRAPH is to redesign IncrementalDBSCAN for an incrementally growing graph. The second issue covered by DENGRAPH is the forgetting of old interactions with help of an ageing function.

The paper is organized as follows. In the next section, we discuss related work on community discovery for static and dynamic environments. In Section 3, we briefly introduce those underpinnings of IncrementalDBSCAN that have been also used in DENGRAPH and then we present DENGRAPH for graph clustering over a stream. In Section 4, we study the behavior of DENGRAPH on a large dataset, the ENRON mail server log. The last section concludes our study. A summary of DENGRAPH and its behavior has appeared in (Falkowski, Barth and Spiliopoulou, 2007).

RELATED WORK

We distinguish between the analysis of community dynamics for *a priori known* communities and for communities that are to be discovered. The former subject is a known, though by no means trivial problem. For example, Backstrom et al. propose a method that uses the underpinnings of spread of influence (Kempe et at., 2003) to analyze the evolution of two *explicitly defined* communities (Backstrom, Huttenloch, Kleinberg and Lan, 2006).

We are interested in discovering and monitoring communities that are *not* known a priori, but rather emerge, evolve, mutate or decline as time passes. Studies in this context emerge only recently (Berger-Wolf and Saia, 2006; Falkowski, Bartelheimer and Spiliopoulou, 2006). In our earlier work on community dynamics (Falkowski et at., 2006), we define a community as a group of cohesive subgroups (by definition of (Girvan and Newman, 2002)) that have been discovered at different time points and exhibit strong similarity to each other. Our method CODYM discovers cohesive subgroups at different time points and then forms communities over time by clustering the subgroups. The basic approach is enhanced by statistics on the evolution of the communities and of the cohesive groups inside them. The authors of Berger-Wolf et al., 2006 define a community over time as a persistent homogeneous formation and propose a method that discovers those formations over a network that remain persistent and homogeneous as time proceeds.

In this paper, we address an orthogonal problem, the *adaptation* of a community to a changing environment. This issue has been studied in the context of stream clustering, where the tasks are (1) to adapt a set of clusters to a stream of records that exhibit drifts and shifts and, orthogonally, (2) to capture and understand the changes that occur in the clusters as the stream drifts/shifts. Advances in this area include, among else, the incremental clustering algorithms of (Aggarwal, Han, Wang and Yu, 2003; Cao, Ester, Qian, Zhou, 2006; Ester, Kriegel, Sander, Wimmer and Xu, 1998; Nasraoui, Cardona Uribe and Rojas Coronel, 2003) and the cluster change detection methods of (Aggarwal, 2005; Bartolini, Ciaccia, Ntoutsi, Patella and Theodoridis, 2004; Ganti, Gehrke and Ramakrishnan, 2000, Spiliopoulou, Ntoutsi, Theodoridis and Schult, 2006). The method proposed by Aggarwal and Yu, 2005 studies community evolution but focuses on change detection rather than community adaptation.

Our work on the adaptation of communities is inspired by research advances on clustering over streams. However, stream clustering algorithms cannot be exploited as they are, since they are designed for sets of records rather than for networks. Our incremental graph mining algorithm DENGRAPH is based on the concepts of the IncrementalDBSCAN clustering algorithm of Ester et al., 1998 but its semantics are adjusted for network analysis.

THE ALGORITHM DENGRAPH

Before introducing our algorithm DENGRAPH, we briefly outline DBSCAN (Ester, Kriegel, Sander, Xu, 1996) and its incremental variation IncrementalDBSCAN (Ester et al., 1998). Then, we transfer the basic concepts of these algorithms to graph mining by first defining proximity for graph nodes. We then present DENGRAPH for graph clustering and subsequently detail on the adaptation of clusters/communities to new interactions and to the elimination of old interactions.

DBSCAN and Incremental DBSCAN

DBSCAN stands for "Density-Based Spatial Clustering of Applications with Noise". For DBSCAN, a cluster is a continuous area of arbitrary shape that is denser than its surroundings. To capture this into a cluster, DBSCAN scans the data points in the dataset and computes neighborhoods. A neighborhood has a given radius (ε) and must contain a minimum number of

points (η). A data point that has such a neighborhood around it is termed a *core point*. A data point that has no such neighborhood is a *noise point*, unless it is itself located in the neighborhood of a core point; then, it is a *border point*. The two thresholds ε, η ensure that neighborhoods are dense areas. DBSCAN builds clusters by connecting adjacent neighborhoods. Informally, the DBSCAN process corresponds to covering an area with overlapping cyclical *tiles*, where a tile has a radius of ε and corresponds to a neighborhood. If a sub area is not dense enough to form a neighborhood, i.e. it has less than η points within radius ε, then no tile can be placed over it. The formal definitions and the complete algorithm can be found in Ester et al., 1996.

In Ester et al., 1998, DBSCAN has been extended to an incremental algorithm. IncrementalDBSCAN considers insertions (new records arrive) and deletions (old records are forgotten) and identifies neighborhoods that are affected by these updates: A neighborhood may have more (resp. less) than η points in it after the arrival of the new record (resp. the removal of the old one) and can thus be covered with a tile (resp. the tile must be removed). These local effects imply that clusters may grow, shrink, merge or split. IncrementalDBSCAN inherits the robustness of DBSCAN towards noisy data and is quite effective in adjusting the clusters.

DBSCAN and IncrementalDBSCAN have been mainly designed for spatial data. For our incremental graph mining algorithm DENGRAPH we transfer the idea of density-based incremental clustering to social network (graph) structures. To do so, we first define a distance function between pairs of vertices in graphs analogously to the distance between points in spatial data sets. We then specify update functions for the incremental changes in the data set.

Defining Proximity over a Graph of Interactions

We first focus on a static graph $G(V, E)$ and discuss its adaptation in the next section. The set of vertices V contains the actors involved in the social network. The set of edges E captures their interactions. An edge between two actors p, q exists only if they interacted at least once during the period of observation. It is undirected but is associated with two values that capture directional semantics: $I_{p,q}$ is the number of interactions that actor p performed towards q, while $I_{q,p}$ is the number of interactions from q to p. It holds that $I_{p,q} + I_{q,p} \geq 1$, otherwise the edge (p,q) would not exist.

The intention of DENGRAPH is to cluster actors into communities. Traditionally, clustering is based on proximity of the objects to be clustered. On a graph of interactions, we model proximity between two actors on the number of their interactions:

Definition 1 *Let $G(V,E)$ be the graph of interactions and let $p,q \in V$ be two vertices/actors. Let $intensity(p,q) = \min(I_{p,q}, I_{q,p})$ be the "intensity" of the interaction between them. Their "proximity" $prox(p,q)$ is defined as:*

$$prox(p,q) = \begin{cases} 1 & p = q \\ 1 - 1/intensity(p,q) & \exists (p,q) \in E \\ undefined & \text{not} \exists (p,q) \in E \end{cases}$$

The proximity function, where defined, ranges in [0, 1] and is symmetric. If only one interaction exists between them, then heir proximity is zero. The proximity value increases asymptotically towards 1 as the number of interactions increases *reciprocally*.

Reciprocity in interaction is of paramount importance here: If p addresses q 1000 times but q addresses p only twice, the proximity of the two will be the same as if p had addressed q only twice. This notion of proximity suppresses the impact of broadcasts and is appropriate for networks where some nodes are characterized by broadcast behavior, e.g. a secretary who regularly sends announcements to all company personnel. Such broadcasts should not be confused with person-to-person interactions.

DENGRAPH, like DBSCAN, is based on the notion of distance rather than proximity. So, we define the following distance function:

Definition 2 *Let $G(V,E)$ be the graph of interactions and let $p,q \in V$ be two interacting actors, i.e. $\exists (p,q) \in E$. Their "distance" is $dist(p,q) := 1 - prox(p,q)$.*

This function is defined only for interacting actors. It is still adequate for our graph clustering task, though, since DENGRAPH does not consider the proximity between arbitrary vertices but rather computes the "ε-neighborhood" of each vertex:

Definition 3 *Let $G(V,E)$ be the graph of interactions and let $p \in V$ be a vertex. The ε-neighborhood of p is the set of vertices that are more proximal to p than ε, i.e. $N_\varepsilon(p) = \{q \in V \mid \exists(p,q) \in E \wedge dist(p,q) \leq \varepsilon\}$.*

Similarly to DBSCAN (Ester et al., 1996), we define the concepts of *core vertex*, *noise vertex* and of *density-reachable* and *density-connected* vertices. However, we do so on the basis of ε -neighborhoods rather than using a distance function over the whole set of vertices:

Definition 4 *A $p \in V$ is a "core vertex" if and only if $\|N_\varepsilon(p)\| \geq \eta$. If $\|N_\varepsilon(p)\| < \eta$, then p is a "noise vertex", unless there is a core vertex q such that $p \in N_\varepsilon(q)$. Then, p is a "border vertex".*

We use the notion of core vertex to define *reachability* among vertices:

Definition 5 *Let $p,q \in V$ be two vertices. p is "directly density-reachable" from q within V with respect to ε, η if and only q is a core vertex and p is in its neighborhood, i.e. $p \in N_\varepsilon(q)$.*

p is "density-reachable" from q within V with respect to ε and η (notation: $p >_V q$) if there is a chain of vertices $p_1,\ldots,p_n$ such that $p_1 = q$ and $p_n = p$ and for each $i = 2,\ldots,n$ it holds that p_i is directly density-reachable from p_{i-1} within V w.r.t. ε and η.

By this definition, a vertex *p* cannot be density-reachable from a vertex *q* unless *q* is a core vertex. This restriction is removed by introducing the notion of *density-connectivity* between vertices, none of which needs to be a core vertex (Ester et al., 1998).

Definition 6 *Let $p,q \in V$ be two vertices. p is "density-connected" to q within V w.r.t. ε and η if and only if there is a vertex $o \in V$ such that $p >_V o$ and $q >_V o$.*

A cluster is a "dense subgroup" composed of all vertices that are density-connected within *V* w.r.t. *ε*, *η*.

Definition 7 *Let $G(V,E)$ be the graph of interactions. A non-empty set $DS \subseteq V$ is a "dense subgroup" w.r.t. ε, η if and only if:*

- *For all $p,q \in V$ it holds that if $p \in DS$ and $q >_V p$, then $q \in DS$ (Maximality condition).*
- *For all $p,q \in DS$ it holds that p is density-connected to q within V w.r.t. ε, η (Connectivity condition).*

To build a cluster, DENGRAPH traverses the graph and places all density-connected points it encounters to the same cluster. If a vertex is not density-connected to the vertices seen thus far, it is assigned to the next cluster candidate. Not each vertex becomes member of a cluster: If a vertex does not have an adequately dense neighborhood w.r.t. *ε*, *η* and is not density-connected to any other vertex, then it is termed a *noise vertex* and its cluster candidate is dropped.

Static Community Discovery

The pseudocode of the DENGRAPH core for community discovery in the static graph is depicted in Algorithm 1: DENGRAPH traverses the graph, processing each vertex in turn. If it is a core vertex, DENGRAPH expands its neighborhood towards further vertices that are density-reachable from it and places them into a stack. This stack contains vertices that are density-connected to each other. They are then assigned to the same cluster, marked by the *cluster id*. The expansion continues until all vertices are popped from the stack, i.e. until no further density-connected vertices can be found. Then DENGRAPH continues with the next non-labeled vertex and builds a new cluster. If a vertex is not a core vertex and has not been assigned to a cluster, then it is marked as noise.

```
Input:  Graph G(V,E), ε, η
Output: Set V with each vertex labeled with its cluster id

Begin
  Repeat
    Select a p∈V that is not yet labeled.
    Compute N_ε(p).
    if (p is a core vertex) then
      Generate a new cluster id and assign it to each
      q∈{p}∪N_ε(p).
      Push q into a stack for p.
      repeat
        Pop the top vertex q of the stack.
        Compute the ε-neighborhood of q.
        if (‖N_ε(q)‖≥η) then
            Label the non-labeled members of N_ε(q) with id and
            push them into the stack for p.
        end
      until (the stack for p is empty)
    end
    if (p is not labeled) then
      Label p as "noise vertex".
    end
  until (all vertices in V are labeled)
end
```

Algorithm 1. Pseudocode of the DENGRAPH core

Incremental Community Adaptation

We have thus far assumed that the graph of interactions is static. In the dynamic scenario, the interactions arrive as a stream: new edges are added, old edges are forgotten. The DENGRAPH core of Algorithm 1 is extended to adapt the clusters incrementally. In the following, we present the encountered types of change and describe the action taken for each. The "forgetting" of old edges reflects the intuitive observation that a community is characterized better by recent interactions rather than from past activities. We model this with an *ageing function f*() which decreases the weights of the interactions seen thus far. An edge is deleted when its weight becomes zero.

Types of Change

The changes to be dealt with by DENGRAPH can be summarized as changes in the *relationships among actors*, namely changes in the intensity of interaction, including the emerging of new interactions (and new actors) and the decay of old ones. These changes influence the distance between vertices and thus the contents of ε-neighborhoods. For example, the intensification of an interaction may make two vertices more proximal to the effect of one entering the neighborhood of the other and making it a core vertex. Then, this neighborhood can result in the expansion of the cluster it belongs to or even become a bridge between two formerly distinct clusters.

DENGRAPH distinguishes between *positive changes* that may lead to community expansion or fusion or to new communities, and *negative changes* that may result to a community split or decay. We discuss the adaptation to these changes below. Since neighborhoods, distances and sets of vertices/edges are time-dependent, we mark each variable and function with the time point *t* to which it refers, e.g. $dist^t(p,q)$ is the distance between *p*, *q* at *t*.

Adaptation to Positive Changes

Let $t, t+1$ be two adjacent time points and let p, q be two actors. Further, assume that $dist^t(p,q)$ were undefined or more than ε and that $dist^{t+1}(p,q) \leq \varepsilon$. The following cases may occur (cf. also Figure 1):

- *New core vertex*: At least one of p, q becomes a core vertex. DENGRAPH processes the new core vertex *p* and its neighbors (including *q*) and forms a neighborhood. It checks whether p, q were already associated with cluster

identifiers. If not, they form a new cluster, otherwise their neighborhood becomes part of an existing cluster or causes the fusion of clusters. If *q* is also a core vertex, DENGRAPH processes it with the neighbors of *p*.

- *New border vertex*: A former noise vertex *p* becomes (directly) density-reachable from a core vertex *q*. DENGRAPH labels *p* with the *id* of the cluster containing *q*.
- *New neighbor*: A new vertex *p* is added but none of the above cases holds. Then DENGRAPH just updates the intensity values of the graph.

Positive changes in the graph of interactions lead to the following adaptations of clusters/communities:

1. *Creation*: A new cluster emerges to accommodate vertices that have become core vertices and have not yet been assigned to a cluster. These vertices appear in the stack of DENGRAPH without a cluster *id* assigned to them; DENGRAPH generates a new cluster *id* and assigns it to them.
2. *Absorption*: A neighborhood of former noise vertices emerges as part of an existing cluster. These vertices appear in the stack of DENGRAPH associated with the *id* of an existing cluster.
3. *Merge*: A new neighborhood is formed, containing core vertices that belong to different clusters. These vertices appear in the stack of DENGRAPH associated with different cluster identifiers. DENGRAPH generates a new cluster *id* and replaces the old identifiers with it, i.e. merges the old clusters into a single new one.

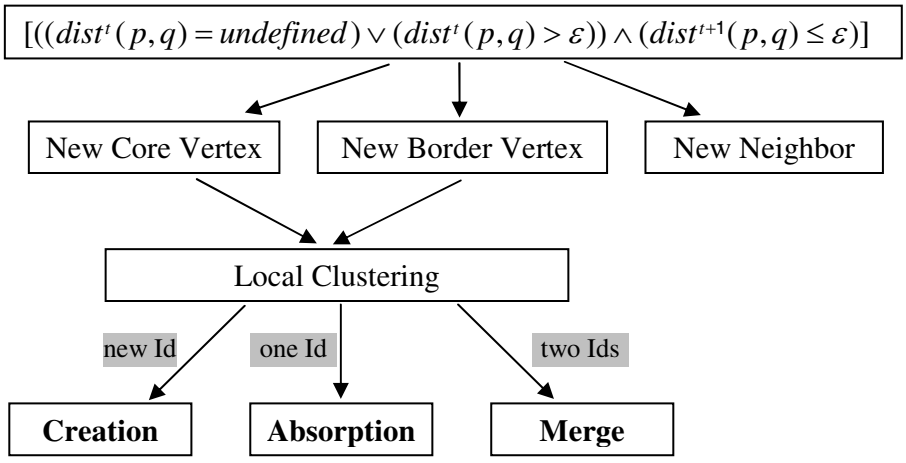

Figure 1. The effects of increased proximity among actors upon the community

Adaptation to Negative Changes

Let $t, t+1$ be two adjacent time points and let p, q be two actors. Further, assume that $dist^t(p,q) \leq \varepsilon$ and that $dist^{t+1}(p,q) > \varepsilon$ or is undefined, i.e. the most recent interactions between p, q are so old that they are forgotten. Then, $q \in N_\varepsilon^t(p)$ and $q \notin N_\varepsilon^{t+1}(p)$. The following cases may occur (cf. also Figure 2):

- *Lost core vertex*: *p* was a core vertex but is no more, because its neighborhood shrinked after the removal of *q*, i.e. $\left\|N_\varepsilon^{t+1}(p)\right\| < \eta$. Hence, the neighborhood of *p* does no more belong to the cluster that contained *p*.
- *Lost edge to core vertex*: *p* was and is still a core vertex, but *q* was a border vertex to *p*, so it may now become noise. Hence, *q* is lost to the cluster that contained *p* and further neighborhoods containing *q* may stop being density-connected to this cluster.
- *Lost edge between cores*: Both p, q were and are still core vertices, but they are no more directly density-reachable from each other. Hence, the cluster containing the two vertices may become split.
- *Lost neighbor*: None of p, q was a core vertex. If they were border vertices, the contents of the affected cluster must be computed. If they were noise vertices, then no cluster is affected, because they did not belong to a cluster anyway.

For each of the above cases except the last sub case, DENGRAPH recomputes the neighborhoods of the affected core vertices. It then recomputes the contents of the clusters to which these neighborhoods belonged and performs one of the following types of cluster adaptation:

1. *Removal*: If there are no more core vertices in the cluster, DENGRAPH removes it and marks its elements as noise vertices.
2. *Reduction*: Some of the core or border vertices of the cluster have become noise but the cluster still has core vertices. So, DENGRAPH marks the affected vertices as noise and the cluster shrinks.
3. *Split*: One or more of the core vertices of a cluster are no more density-connected to it but are still core vertices. DENGRAPH assigns them to a new cluster, i.e. the original cluster is split.
4. *Move*: Some of the core vertices of the cluster have become border vertices for another cluster. DENGRAPH moves them by assigning the *id* of the latter cluster to them.

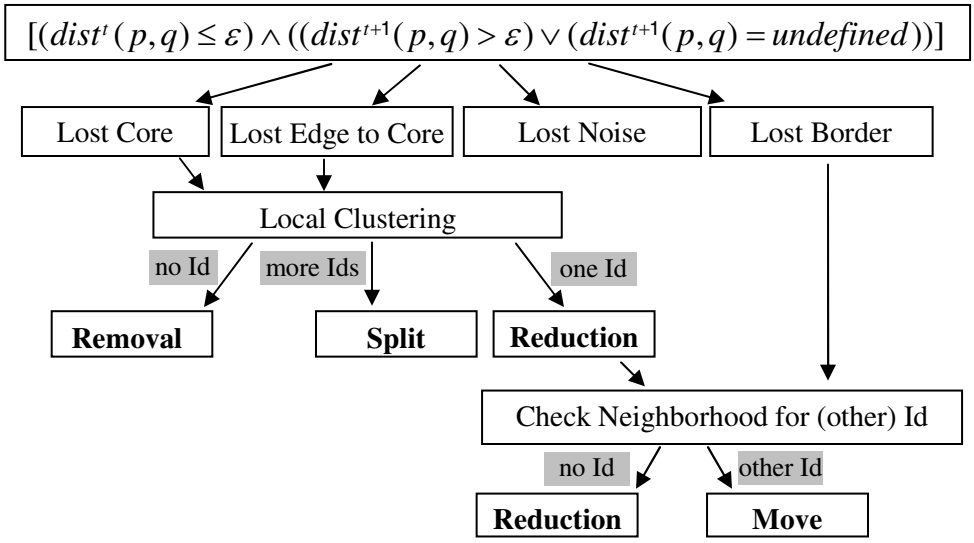

Figure 2. The effects of decreased proximity or actor/edge removal

Summary of DENGRAPH

Summarizing, DENGRAPH takes as input a sequence of interactions from the evolving graph and computes dense subgroups on the basis of interaction intensity. These subgroups are the clusters or communities and are adapted to the positive and negative changes described above. As old interactions are forgotten, DENGRAPH may split the communities, shrink them, move members to other communities or even delete them. As new interactions arrive, DENGRAPH creates new emerging communities, expands existing ones and occasionally merges formerly distinct communities to a new one.

EXPERIMENTS

We applied the widely used edge betweenness clustering algorithm and our DENGRAPH to a social network graph - the ENRON email data set - and compared both results to obtain information about the characteristics of both clustering methods.

Understanding the Data Set

In the following, we present the data set that we used for our experiments, describe its basic characteristics and discuss the determination of the parameters for DENGRAPH.

The Data Set

For the experiments we used the ENRON email data set provided by University of Southern California [http://www.isi.edu/~adibi/Enron/Enron.htm]. The data set contains emails from end of 1998 to mid 2002. Since the number of messages differs considerably at the beginning and at the end we use the interval October 1999 to March 2002. In this time

period 248.353 distinct messages to 2.046.843 recipients were sent. We model a graph from the email data, where actors are represented as vertices and the email exchange between actors is modeled as edges. The weight of the edges (distance) is calculated according to Definition 2.

As expected, the data set shows social network characteristics: a right-skewed degree distribution (see Figure 3 – left side) and a short average distance between vertices (small-world effect) (see Figure 3 - right side). Due to its size and these characteristics, it is particularly suitable for dense subgraph detection in social networks.

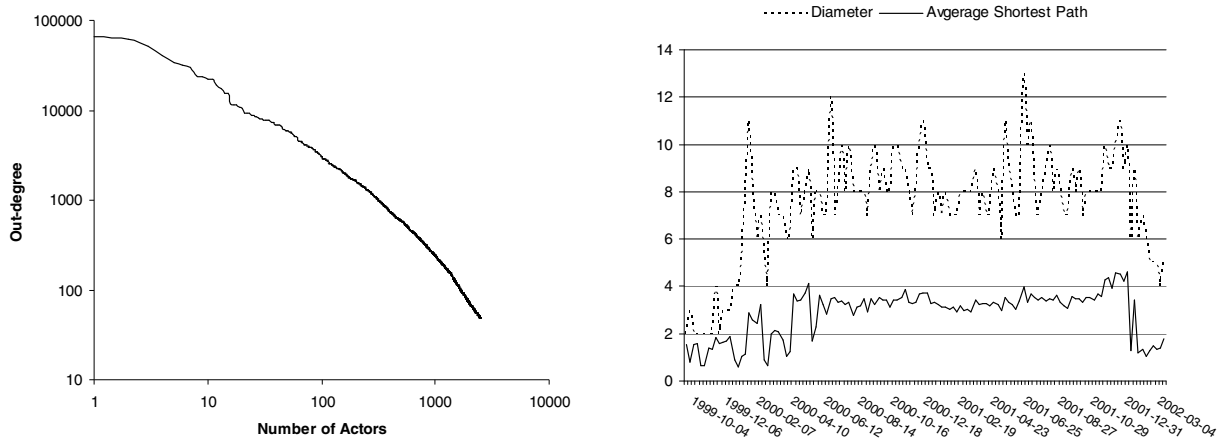

Figure 3. Left: Out-degree distribution for all actors with an out-degree ≥50 (log-log plot). Right: Diameter and average shortest path for seven-day intervals (October 1999 to March 2002)

In the course of a week about 60 percent of all actors send one to two messages to their neighbors ($N_\varepsilon(p)$). Only a few actors exchange three are more messages in one week. Thus, almost 40 percent of all actors do not send emails on a regular basis. These actors are noise objects, since they are not considered as members of a subgroup due to their (long-term) inactivity. Thus, a lot of the data cannot be clustered and is labeled as noise.

This characteristic of the data set poses a severe performance problem for the edge betweenness algorithm: The algorithm is a hierarchical divisive clustering algorithm and obtains clusters by deleting edges that are bridges between clusters. For this, in each cluster iteration, the edge betweenness value for every edge must be calculated and the edge with the highest betweenness is removed. If the graph contains many vertices that are only weakly connected, the calculation of the edge betweenness value is computationally expensive. Therefore, a data preparation to remove rather isolated vertices is necessary to improve the performance of the edge betweenness algorithm.

Determining Parameters ε and η for DENGRAPH

To determine the parameters ε and η for the experiments we analyzed the email data over a three-months-period in 2001 to see how the size of each vertice's neighborhood and the percentage of noise depend on a given η. As shown in Table 1, the percentage of noise increases dramatically with increasing η (minimum number of points in the neighborhood without considering ε). When choosing η = 2 on average 60 percent of all vertices are noise and do not belong to any subgroup. For η = 3 it is 77 percent and for η = 4, 85 percent.

In Table 2 it is shown, that the average distance of the neighbors varies only slightly depending on η. The distances vary between 0.55 and 0.64. This shows that the distance between actors does not necessarily depend on the number of neighbors: Even actors with large neighborhoods maintain close relationships to their neighbors. Note that the average distance of actors is only calculated for those actors who actually have η neighbors.

For different ε values we analyzed the average number of messages that are exchanged between vertices in each interval. The numbers fluctuate noticeable during the year as expected: We observe high traffic periods and intervals with rather few interactions (e.g. summer vacation time). Based on the analysis of the data set characteristics we chose the parameters for the experiments as follows: η = 3 and $\varepsilon = 1/3$.

Interval	η=1	η =2	η =3	η =4	η =5	η =6
02.-08.10.2001	28.67	60	77.67	83	87.67	92.33
09.-15.10.2001	28.36	59.94	74.56	81.29	87.43	91.23
16.-22.10.2001	21.17	50.12	68.13	77.13	85.64	90.02
23.-29.10.2001	15.58	46.95	66.14	77.2	83.07	87.36
30.10.-05.11.2001	27.91	58.59	74.54	83.74	88.34	91.72
06.-12.11.2001	29.71	61.47	79.12	88.82	92.06	95
13.-19.11.2001	18.23	46.84	62.28	75.44	84.81	87.85
20.-26.11.2001	25.91	56.55	74.65	83.01	87.19	89.69
27.11.-03.12.2001	31.45	63.84	77.04	83.65	89.62	91.82
04.-10.12.2001	38.94	71.15	87.5	92.79	95.67	97.6
11.-17.12.2001	33.64	67.27	84.55	90.91	95	96.36
18.-24.12.2001	34.22	71.12	89.3	95.19	96.79	97.33
25.-31.12.2001	37.3	72.22	84.92	92.06	96.83	97.62
Mean	28.55	60.47	76.95	84.94	90.01	92.76

Table 1. Percentage of noise for a given η

Interval	η=1	η =2	η =3	η =4	η =5	η =6
02.-08.10.2001	0.55	0.60	0.58	0.66	0.68	0.62
09.-15.10.2001	0.53	0.62	0.66	0.70	0.72	0.75
16.-22.10.2001	0.46	0.57	0.58	0.60	0.57	0.55
23.-29.10.2001	0.48	0.54	0.57	0.58	0.60	0.62
30.10.-05.11.2001	0.56	0.63	0.69	0.70	0.69	0.72
06.-12.11.2001	0.58	0.71	0.72	0.71	0.77	0.77
13.-19.11.2001	0.48	0.57	0.64	0.67	0.70	0.77
20.-26.11.2001	0.51	0.58	0.60	0.58	0.68	0.73
27.11.-03.12.2001	0.58	0.59	0.64	0.65	0.70	0.70
04.-10.12.2001	0.65	0.74	0.75	0.74	0.76	0.70
11.-17.12.2001	0.56	0.65	0.69	0.66	0.76	0.75
18.-24.12.2001	0.68	0.74	0.75	0.81	0.72	0.80
25.-31.12.2001	0.63	0.72	0.72	0.66	0.58	0.61
Mean	0.55	0.62	0.64	0.62	0.61	0.62

Table 2. Average distance of η-nearest neighbor (for actors who have a η-nearest neighbor)

Experimental Setting

For the edge betweenness clustering we clustered the aggregated email data of one period (seven days). To determine the strength (distance) of the tie between two actors, we used the number of reciprocal interactions. The clustering results can be represented as a dendrogram and we cut the dendrogram at the level with the highest modularity as proposed by Girvan and Newman, 2002.

For the incremental DENGRAPH, we first cluster the email data for a period of seven days. Then the clustering is updated by inserting all interactions that are observed on the next day and the data of the first day of the interval is deleted. Thus, days 1-7 are clustered, afterwards day 8 is inserted and all data from day 1 is deleted. The obtained clustering after inserting the last day of one period is used for the comparison with the edge betweenness clustering.

Comparison EBC and DENGRAPH

We performed a clustering on 130 intervals (one interval comprises one week) with the weighted edge betweenness clustering (EBC) and with DENGRAPH. Since a comparison to a "real clustering" is not possible for this data set, we quantitatively compare both clustering results and discuss basic characteristics. The EBC shows about 12,000 subgroups; 81 percent are singletons (groups with only one member). 40 percent of the remaining actors are grouped in subgroups with more than 50 members. On average, 18 groups are built per interval with on average 10 members. On average one group per interval has more than 100 members and the average group size is very low due to the large amount of singletons. In summary, one could say that even though the EBC reveals a high number of subgroups, most contain only one member. The other 20 percent tend to be members of rather large groups.

For $\varepsilon = 1/3$ and $\eta = 3$, DENGRAPH classifies on average 93 percent of all vertices as noise. If we compare the subgroups of DENGRAPH with the EBC subgroups (with more than one member) we obtain the following: DENGRAPH subgroups are in 39 percent of all cases subsets of larger EBC subgroups. On average, three DENGRAPH subgroups form one EBC subgroup. DENGRAPH thus tends - due to its density-based notion of a subgroup - to find smaller subgroups that are more closely connected. Since most subgroups detected with DENGRAPH have an (large) overlap with the EBC groups – most of them are subsets of the groups that EBC finds – we conclude that both methods in general reveal similar community structures, however with a different granularity.

In Table 3 an overview of the main characteristics of both clustering method is shown.

	Percentage of Singleton-Clusters (Clusters w/ one member)	**Average Number of Clusters per Interval**	**Average Cluster Size**
Edge Betweenness Clustering	81 %	18	10
DENGRAPH	93 % (not clustered as singletons but labeled as noise vertices)	2	7

Table 3. Comparison Edge Betweenness Clustering and DENGRAPH

DENGRAPH reveals smaller more dense groups whereas EBC tends to merge also less close actors. Even though a more thorough investigation regarding the time complexity needs to be performed, we can already state that the density-based approach outperforms the hierarchical methods by an order of magnitude. This is in particular true for large data sets with noise as the ENRON email data.

The choice of the algorithm, thus, mainly depends on the size of the dataset (namely the number of vertices) and the notion of what a cluster constitutes in the respective application. DENGRAPH is especially suited for large, noisy datasets since it detects dense, local neighborhoods and is capable to efficiently remove outliers. A cluster is then a set of dense neighborhoods. The members of a cluster must therefore not necessarily be densely connected with all other members of the cluster, but to the members in its neighborhood. With EBC, clusters are built where all members within the cluster are more densely connected compared to the number of interactions with members outside the cluster (intra-cluster density vs. inter-cluster sparsity). Therefore, besides the size of the data set, the notion of what a community constitutes is relevant for the choice of the algorithm.

CONCLUSIONS

In this paper we presented an incremental density based algorithm to detect dense subgroups in (social) networks. We adapted the IncrementalDBSCAN algorithm to graph structures: We defined a distance function for two interacting vertices and update functions for the incremental clustering. First experiments with the ENRON data set and comparisons with results from the edge betweenness clustering show, that the algorithm reveals meaningful subgroups. The EBC tends to group on the one hand rather less dense vertices to big subgroups and on the other hand many singletons and small groups. DENGRAPH is capable to efficiently remove small groups as noise and has the characteristic to detect smaller more dense structures compared to the EBC. Based on these first experiments it can be summarized that DENGRAPH is - compared to the edge betweenness algorithm - a very efficient algorithm for dense subgraph detection in social networks.

REFERENCES

1. Aggarwal, C. C. (2005) On Change Diagnosis in Evolving Data Streams*, IEEE Transactions on Data and Knowledge Engineering*, 17(5), 587-600.
2. Aggarwal, C. C., Han, J., Wang, J. and Yu, P. S. (2003) A Framework for Clustering Evolving Data Streams, *Proceedings of VLDB 2003*, 81-92.
3. Aggarwal, C. C. and Yu, P. S. (2005) Online Analysis of Community Evolution in Data Streams, *Proceedings of SIAM International Data Mining Conference*.
4. Backstrom, L., Huttenlocher, D., Kleinberg, J. and Lan, X. (2006) Group Formation in Large Social Networks: Membership, Growth, and Evolution, *Proceedings of KDD 2006*, 44-54.
5. Berger-Wolf, T. and Saia, J. (2006) A Framework for Analysis of Dynamic Social Networks, *Proceedings of KDD 2006*, 523-528.
6. Bartolini, I., Ciaccia, P., Ntoutsi, I., Patella, M. and Theodoridis, Y. (2004) A unified and flexible framework for comparing simple and complex patterns, *Proceedings of ECML/PKDD 2004*, 496-499.
7. Cao, F., Ester, M., Qian, W. and Zhou, A. (2006) Density-based Clustering over an Evolving Data Stream with Noise, *Proceedings of SIAM International Data Mining Conference 2006*, 326-337.
8. Domingos, P. and Richardson, M. (2001) Mining the Network Value of Customers, In: *Proceedings of KDD 2001*, 57-66.
9. Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M. and Xu, X. (1998) Incremental Clustering for Mining in a Data Warehouse Environment, *Proceedings of 24th VLDB Conference*, 323-333.
10. Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proceedings of KDD 1996*, 226-231.
11. Falkowski, T., Bartelheimer, J. and Spiliopoulou, M. (2006) Mining and Visualizing the Evolution of Subgroups in Social Networks, *Proceedinsg of 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI-06)*, 52-58.
12. Falkowski, T., Barth, A. and Spiliopoulou, M. (2007) DENGRAPH: A Density-based Community Detection Algorithm, *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI-07)*, 112-115.
13. Ganti, V., Gehrke, J. and Ramakrishnan, R. (2000) Demon: Mining and monitoring evolving data, *Proceedings. of ICDE 2000*, 439-448.
14. Girvan, M. and Newman, M. E. J. (2002) Community structure in social and biological networks*, PNAS*, 99(12), 7821-7826.
15. Kempe, D., Kleinberg, J. and Tardos, É. (2003) Maximizing the spread of influence through a social network, *Proceedings of KDD 2003*, 137-146.
16. Nasraoui, O., Cardona Uribe, C., Rojas Coronel, C. and González, F. A. (2003) Tecno-streams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model, *Proceedings of ICDM 2003*, 235-242.
17. Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y. and Schult, R. (2006) MONIC - Modeling and Monitoring Cluster Transitions, *Proceedings of KDD 2006*, 706-711.